CLAIMS

What is claimed is:

1. A method for arithmetic expression optimization, comprising:

   receiving a first instruction defined for a first processor having a first base, said instruction comprising an operator and at least one operand;

   converting said first instruction to a second instruction optimized for a second processor having a second base when said at least one operand does not carry potential overflow beyond said second base or when said operator is insensitive to overflow, said second base smaller than said first base; and

   converting instructions in a chain of instructions to a wider base when said at least one operand carries the potential for overflow beyond said second base and when said operator is sensitive to overflow, said chain bounded by said second instruction and a third instruction that is the source of potential overflow associated with said at least one operand, said third instruction having been previously optimized, said wider base larger than said second base and smaller or equal to said first base.

2. The method of claim 1 wherein said first instruction is arithmetic.

3. The method of claim 1 wherein said first instruction comprises a non-arithmetic, type-sensitive instruction.

4. The method of claim 1, further comprising repeating said receiving, said converting a first instruction and said converting instructions in a chain of instructions for instructions that comprise a program.

5. The method of claim 1, further comprising linking each instruction to input instructions in all control paths.

6. The method of claim 1 wherein

said first instruction is defined for a first processor having a first base; and

said second instruction is defined for a second processor having a second base;

7. The method of claim 6 wherein

said first processor comprises a Java™ Virtual Machine; and

said second processor comprises a Java Card™ Virtual Machine.

8. The method of claim 6 wherein

said first processor comprises a 32-bit processor; and

said second processor comprises a resource-constrained 16-bit processor.

9. The method of claim 1 wherein

said third instruction is not a source instruction; and

said converting instructions in a chain of instructions further comprises:

recursively examining input instructions until said third instruction is obtained; and setting the type of said third instruction to equal said second type.

10. The method of claim 1 wherein

said third instruction comprises a source instruction; and

said converting instructions in a chain of instructions further comprises:

recursively examining input instructions until said third instruction is obtained;

setting the type of said third instruction to equal said second type; and

repeating said changing for each input instruction of said third instruction.

11. The method of claim 1, further comprising recording conversion results, said recording comprising:

determining potential overflow associated with said second instruction; and

generating an output stack based at least in part on execution of said second instruction.

12. The method of claim 11 wherein said determining further comprises:

indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, and if said second instruction creates potential overflow; and

indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, if said second instruction does not create potential overflow, if said second instruction propagates potential overflow, and if at least one operand in said at least one input stack has potential overflow.

13. The method of claim 11 wherein said generating further comprises:

creating a new output stack based at least in part on one of said at least one input stack;

updating said new output stack based at least in part on operation of said second instruction; and

indicating another instruction conversion pass is required if said new stack does not equal a previous output stack.

14. A method for arithmetic expression optimization, comprising:

step for receiving a first instruction defined for a first processor having a first base, said instruction comprising an operator and at least one operand;

step for converting said first instruction to a second instruction optimized for a second processor having a second base when said at least one operand does not carry potential overflow beyond said second base or when said operator is insensitive to overflow, said second base smaller than said first base; and

step for converting instructions in a chain of instructions to a wider base when said at least one operand carries the potential for overflow beyond said second base and when said operator is sensitive to overflow, said chain bounded by said second instruction and a third instruction that is the source of potential overflow associated with said at least one operand, said third instruction having been previously optimized, said wider base larger than said second base and smaller or equal to said first base.

15. The method of claim 14 wherein said first instruction is arithmetic.

16. The method of claim 14 wherein said first instruction comprises a non-arithmetic, type-sensitive instruction.

17. The method of claim 14, further comprising step for repeating said receiving, said converting a first instruction and said converting instructions in a chain of instructions for instructions that comprise a program.

18. The method of claim 14, further comprising step for linking each instruction to input instructions in all control paths.

19. The method of claim 14 wherein

said first instruction is defined for a first processor having a first base; and

said second instruction is defined for a second processor having a second base;

20. The method of claim 19 wherein

said first processor comprises a Java™ Virtual Machine; and

said second processor comprises a Java Card™ Virtual Machine.

21. The method of claim 19 wherein

said first processor comprises a 32-bit processor; and

said second processor comprises a resource-constrained 16-bit processor.

22. The method of claim 14 wherein

said third instruction is not a source instruction; and

said step for converting instructions in a chain of instructions further comprises:

step for recursively examining input instructions until said third instruction is obtained; and

step for setting the type of said third instruction to equal said second type.

23. The method of claim 14 wherein

said third instruction comprises a source instruction; and

said step for converting instructions in a chain of instructions further comprises:

step for recursively examining input instructions until said third instruction is obtained;

step for setting the type of said third instruction to equal said second type; and

step for repeating said changing for each input instruction of said third instruction.

24. The method of claim 14, further comprising step for recording conversion results, said recording comprising:

step for determining potential overflow associated with said second instruction; and

step for generating an output stack based at least in part on execution of said second instruction.

25. The method of claim 24 wherein said step for determining further comprises:

step for indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, and if said second instruction creates potential overflow; and

step for indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, if said second instruction does not create potential overflow, if said second instruction propagates potential overflow, and if at least one operand in said at least one input stack has potential overflow.

26. The method of claim 24 wherein said step for generating further comprises:

step for creating a new output stack based at least in part on one of said at least one input stack;

step for updating said new output stack based at least in part on operation of said second instruction; and

step for indicating another instruction conversion pass is required if said new stack does not equal a previous output stack.

27. A program storage device readable by a machine, embodying a program of instructions executable by the machine to perform a method for arithmetic expression optimization, the method comprising:

receiving a first instruction defined for a first processor having a first base, said instruction comprising an operator and at least one operand;

converting said first instruction to a second instruction optimized for a second processor having a second base when said at least one operand does not carry potential overflow beyond said second base or when said operator is insensitive to overflow, said second base smaller than said first base; and

converting instructions in a chain of instructions to a wider base when said at least one operand carries the potential for overflow beyond said second base and when said operator is sensitive to overflow, said chain bounded by said second instruction and a third instruction that is the source of potential overflow associated with said at least one operand, said third instruction having been previously optimized, said wider base larger than said second base and smaller or equal to said first base.

28. The program storage device of claim 27 wherein said first instruction is arithmetic.

29. The program storage device of claim 27 wherein said first instruction comprises a non-arithmetic, type-sensitive instruction.

30. The program storage device of claim 27, said method further comprising repeating said receiving, said converting a first instruction and said converting instructions in a chain of instructions for instructions that comprise a program.

31. The program storage device of claim 27, said method further comprising linking each instruction to input instructions in all control paths.

32. The program storage device of claim 27 wherein

said first instruction is defined for a first processor having a first base; and

said second instruction is defined for a second processor having a second base;

33. The program storage device of claim 32 wherein

said first processor comprises a Java™ Virtual Machine; and

said second processor comprises a Java Card™ Virtual Machine.

34. The program storage device of claim 32 wherein

said first processor comprises a 32-bit processor; and

said second processor comprises a resource-constrained 16-bit processor.

35. The program storage device of claim 27 wherein

said third instruction is not a source instruction; and

said converting instructions in a chain of instructions further comprises:

recursively examining input instructions until said third instruction is obtained; and

setting the type of said third instruction to equal said second type.

36. The program storage device of claim 27 wherein

said third instruction comprises a source instruction; and

said converting instructions in a chain of instructions further comprises:

recursively examining input instructions until said third instruction is obtained;

setting the type of said third instruction to equal said second type; and repeating said changing for each input instruction of said third instruction.

37. The program storage device of claim 27, said method further comprising recording conversion results, said recording comprising:
determining potential overflow associated with said second instruction; and
generating an output stack based at least in part on execution of said second instruction.

38. The program storage device of claim 37 wherein said determining further comprises:
indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, and if said second instruction creates potential overflow; and
indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, if said second instruction does not create potential overflow, if said second instruction propagates potential overflow, and if at least one operand in said at least one input stack has potential overflow.

39. The program storage device of claim 37 wherein said generating further comprises:
creating a new output stack based at least in part on one of said at least one input stack;
updating said new output stack based at least in part on operation of said second instruction; and
indicating another instruction conversion pass is required if said new stack does not equal a previous output stack.

40. An apparatus for arithmetic expression optimization, comprising:

means for receiving a first instruction defined for a first processor having a first base, said instruction comprising an operator and at least one operand;

means for converting said first instruction to a second instruction optimized for a second processor having a second base when said at least one operand does not carry potential overflow beyond said second base or when said operator is insensitive to overflow, said second base smaller than said first base; and

means for converting instructions in a chain of instructions to a wider base when said at least one operand carries the potential for overflow beyond said second base and when said operator is sensitive to overflow, said chain bounded by said second instruction and a third instruction that is the source of potential overflow associated with said at least one operand, said third instruction having been previously optimized, said wider base larger than said second base and smaller or equal to said first base.

41. The apparatus of claim 40 wherein said first instruction is arithmetic.

42. The apparatus of claim 40 wherein said first instruction comprises a non-arithmetic, type-sensitive instruction.

43. The apparatus of claim 40, further comprising means for repeating said receiving, said converting a first instruction and said converting instructions in a chain of instructions for instructions that comprise a program.

44. The apparatus of claim 40, further comprising means for linking each instruction to input instructions in all control paths.

45. The apparatus of claim 40 wherein

said first instruction is defined for a first processor having a first base; and

said second instruction is defined for a second processor having a second base;

46. The apparatus of claim 45 wherein

said first processor comprises a Java™ Virtual Machine; and

said second processor comprises a Java Card™ Virtual Machine.

47. The apparatus of claim 45 wherein

said first processor comprises a 32-bit processor; and

said second processor comprises a resource-constrained 16-bit processor.

48. The apparatus of claim 40 wherein

said third instruction is not a source instruction; and

said means for converting instructions in a chain of instructions further comprises:

means for recursively examining input instructions until said third instruction is obtained; and

means for setting the type of said third instruction to equal said second type.

49. The apparatus of claim 40 wherein

said third instruction comprises a source instruction; and

said means for converting instructions in a chain of instructions further comprises:

means for recursively examining input instructions until said third instruction is obtained;

means for setting the type of said third instruction to equal said second type; and

means for repeating said changing for each input instruction of said third instruction.

50. The apparatus of claim 40, further comprising means for recording conversion results, comprising:

means for determining potential overflow associated with said second instruction; and

means for generating an output stack based at least in part on execution of said second instruction.

51. The apparatus of claim 50 wherein said means for determining further comprises:

means for indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, and if said second instruction creates potential overflow; and

means for indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, if said second instruction does not create potential overflow, if said second instruction propagates potential overflow, and if at least one operand in said at least one input stack has potential overflow.

52. The apparatus of claim 50 wherein said means for generating further comprises:

means for creating a new output stack based at least in part on one of said at least one input stack;

means for updating said new output stack based at least in part on operation of said second instruction; and

means for indicating another instruction conversion pass is required if said new stack does not equal a previous output stack.

53. A method of using an application software program including arithmetic expression optimization of at least one instruction targeted to a processor, the method comprising:

receiving the software program on said processor, said software program optimized according to a method comprising:

receiving a first instruction defined for a first processor having a first base, said instruction comprising an operator and at least one operand;

converting said first instruction to a second instruction optimized for a second processor having a second base when said at least one operand does not carry potential overflow beyond said second base or when said operator is insensitive to overflow, said second base smaller than said first base; and converting instructions in a chain of instructions to a wider base when said at least one operand carries the potential for overflow beyond said second base and when said operator is sensitive to overflow, said chain bounded by said second instruction and a third instruction that is the source of potential overflow associated with said at least one operand, said third instruction having been previously optimized, said wider base larger than said second base and smaller or equal to said first base; and

executing said at least one instruction on said processor.

54. A smart card having a microcontroller embedded therein, said microcontroller configured to execute a virtual machine, the virtual machine capable of executing a software application comprising a plurality of previously optimized instructions, the instructions optimized by a method comprising:

receiving a first instruction defined for a first processor having a first base, said instruction comprising an operator and at least one operand;

converting said first instruction to a second instruction optimized for a second processor having a second base when said at least one operand does not carry potential overflow beyond said second base or when said operator is insensitive to overflow, said second base smaller than said first base; and

converting instructions in a chain of instructions to a wider base when said at least one operand carries the potential for overflow beyond said second base and when said operator is sensitive to overflow, said chain bounded by said second instruction and a third instruction that is the source of potential overflow associated with said at least one operand, said third instruction having been previously optimized, said wider base larger than said second base and smaller or equal to said first base.